L6: Entry 1 of 1 File: USPT Jan 9, 2007

DOCUMENT-IDENTIFIER: US 7162467 B2
TITLE: Systems and methods for managing distributed database resources

PRIOR-PUBLICATION:

DOC-ID	DATE
US 20020116457 A1	August 22, 2002

Description Paragraph (26):
In addition to the conventional functionality described above, modified JDBC client driver 411 comprises new functionality via resource abstraction layer (RAL) 416. RAL 416 translates API calls into network protocol encapsulated RSU requests. As shown in FIG. 4, RAL 416 interfaces with JDBC connection module 415 to send and receive database requests for processing on RSU 420. RAL 416 includes programming logic for identifying an appropriate RSU to fulfill any data requests or update transactions received by application server 410 from a client application. RAL 416 pools RSUs according to database or application server administrator-derived policies. Such allocation policies may include, for example, a stateful priority queue with multiple access, round robin using low priority RSUs, or geographical (e.g., server name or IP domain name) allocation. Additionally, RAL 416 could use all allocation policies simultaneously to maximize efficient resource allocation or increase service level guarantees for designated application servers. After determining which RSU should receive the database request, RAL 416 sends the request to the RSU for further processing.

Description Paragraph (57):
The complete transaction process is shown in FIG. 6B, using timelines again to visualize the actions of the participants. The application server sends a message to the RSU to open the transaction in step 630, and then the RSU places a new entry in a local Update Request Queue in step 632. It also sends a request for an update to the DSM in step 632, which opens a remote transaction with serializable transaction isolation in step 634. The DSM then initiates an update to the RSU, where the data required for the update are read within the remote transaction in step 636, which ensures proper time synchronization.

Col. 15 lines 1-11

Description Paragraph (58):
The two steps within the dashed box 638 are done as an atomic, serialized operation on the RSU. This can be accomplished using a simple FIFO (First In First Out) queue to hold the update requests, and when each update request is processed, both steps are completed before de-queuing the request. In addition, step 638 is processed after all previously queued update requests. The update step 638 begins by receiving the updates from the DSM. It initiates a separate, concurrent process that starts to implement the updates into the Cache database (step 640). It then scans the update message from the DSM for predicates that are contained in the cache database and marks each of those to prevent their use by the transaction associated with the update request. It then opens a local transaction with serializable isolation, which completes step 638. After step 638 is completed, the remote transaction and the local transaction are synchronized to the same point in

time, and all read and write requests to the Cache database and to the DBMS server will conform to the required serializable transaction isolation level.

CLAIMS:

1. A distributed database caching system for processing transactions between an application server and a central DBMS server, the system comprising: a resource abstraction layer on the application server; and a remote server unit in communication with the application server and a database subscription manager, wherein the remote server unit includes a cache DBMS server to manage a cache database, and wherein the database subscription manager is in communication with the central DBMS server, wherein the application server sends queries for a plurality of users to the remote server unit via the resource abstraction layer, wherein the remote server unit processes each query through the cache DBMS server, and wherein the cache DBMS server checks a data structure consisting of subscribed query predicates, and wherein, if the query is contained within prior query predicates, the remote server unit sends local query results derived from the cache database to the application server, and wherein if the query is not contained within subscribed predicates, the remote server unit sends the query to the data subscriber manager, wherein the database subscription manager retrieves a result from the central DBMS server, and wherein the database subscription manager derives query results from the central DBMS server, and wherein the database subscription manager sends the query results to the remote server unit, and creates a subscription to the query predicate on behalf of the remote server unit, wherein the query results are added to the cache database and the query predicate is added to the query predicate data structure on the remote server unit, completing a subscription to that query predicate.

12. A method of processing a transaction between an application server and a central DBMS server, the method comprising: sending the transaction from the application server to a remote server unit, wherein the transaction is sent via a resource abstraction layer on the application server; determining whether a local result exists in a cache database on the remote server unit; sending the local result from the remote server unit to the application server if the local result exists; sending the transaction from the remote server unit to a data subscriber manager if the local result does not exist, wherein the database subscription manager retrieves a result from the central DBMS server; deriving on the database subscription manager a plurality of predicates from a plurality of transactions processed by the central DBMS server; sending the plurality of predicates from the database subscription manager to the remote server unit; and updating the cache database according to the plurality of predicates.

13. A method of implementing serializable transaction isolation in a distributed database caching system in communication with a central DBMS server, the method comprising; opening a serializable transaction on an application server; placing an entry into a local update request queue on a remote server unit; opening a remote transaction on a central DBMS server using a database subscription manager in response to an update request from the remote server unit; sending a plurality of updates from the database subscription manager to the remote server unit using data obtained from the central DBMS server; processing the plurality of updates in a serialized manner on the remote server unit by checking the plurality of updates to identify predicates in the cache database, locking the predicates in the cache database, and starting a local transaction on a remote server unit; thereby synchronizing the remote transaction and the local transaction.

Previous Doc Next Doc Go to Doc#